APPENDIX - C

BINDING AND RCP ARCHITECTURE :

Consider an Rcp Gate G1, with the following configuration :

{Q1, QA1[8], QA2[8]} → G1{N1[2], L1} → {QA3[8], QA4[8]}

It may be noted that during binding, the Rcp gate G1 selects a set of input queues like :

QA1(2), QA2(2) or
QA1(5), QA2(5)

Where the number within the parenthesis is called the input queue index, and is same for all input queue arrays. The same concept holds for output queue arrays, and the output queues are identified by output queue index like,

QA3(4), QA4(4) or
QA3(7), QA4(7)

Thus binding uses simple association to identify the queues within the queue arrays. However there are many special cases in binding, which are discussed below.

It may be noted that in the above example, Rcp Gate G1 is controlling two invocations of node function N1. Assuming that both invocations bound at the same time, we can write the bindings as :

{Q1, QA1(2), QA2(2)} → N1(1) → { QA3(4), QA4(4)} and
{Q1, QA1(5), QA2(5)} → N1(2) → { QA3(7), QA4(7)}

The important point in the above example is that queue Q1 is bound to both invocations of node function N1. This is a feature of Rcp architecture, where different invocations get different queues from queue arrays, whereas they share the queues. It may be noted that queues can be specified only on the input side of the Rcp gates, and on output side every invocation gets its own set of queues.

Now let us consider a slightly more complex situation, where we have two Rcp gates G1 and G2, and let us assume the following configurations for G1 and G2.

{Q1, QA1[8], QA2[8]} → G1{N1[2], L1} → {QA3[8], QA4[8]}
{Q1, QA1[8], QA2[8]} → G2{N2[2], L2} → {QA5[8], QA5[8]}

In this case the two invocations of N1 and the two invocations of N2 together share the Queue Q1, whereas one invocation of N1 and one invocation of N2 will share queues contained in queue arrays.

For example, QA1(5) is read by either N1(0) or N1(1) and
QA1(5) is also read by either N2(0) or N2(1)

It may be noted that if queue Q1 is of type "input-output", then it can be updated by all four invocations at the same time.

It is possible to have the following configuration :
{Q1, QA1[8], QA2[8]} → G1{N1[2], L1} → {QA3[8], QA4[8]}
{Q1, QA1[8], QA2[8]} → G2{N2[2], L1} → {QA3[8], QA4[8]}

It may be noted that G1 and G2 are now connected to the same local ring L1. The only restriction is that consumers should not use QA3 or QA4 with some other queue array say QA5, however they can use QA3, and QA4 together or independently.